POSTER: CipherLocker: Encrypted File Sharing with Ranked Search <https://cipherlocker.com>

Jan Kasiak, Bogdan Carbutar, Jake Christensen,
Maria Lyukova, Sumeet Bajaj, Mike Boruta,
Radu Sion
Private Machines Inc.
Brooklyn, NY, USA
{jan,bogdan,jake,masha,sumeet,mike,radu}@privatemachines.com

Viorel Popescu, Alex Sorodoc, Gabriel Stan
Privacy Systems SRL
Bucharest, Romania
{viorel,alex,gabriel}@privacysystems.eu

ABSTRACT

Today's (predominantly cloud-based) file sharing products leave users at the mercy of providers and nation-state adversaries with subpoena and National Security Letter (NSL) powers. In-transit and provider-side at-rest encryption do little to handle this. Almost-weekly breaches [7–13, 17] and NSL revelations [2] show that the problem becomes only worse with increasingly privacy-unfriendly regulation [14]. We believe it is important to provide hype-free, easy-to-use strongly-secure solutions that protect individual privacy while also defeating cloud breaches and compromises.

CipherLocker provides practical, easy-to-use, client-side encrypted file sharing with integrated ranked search. All data and metadata are strongly encrypted before leaving the client. Users can securely store, share, sync, and search. The design does not allow even a compromised or compelled cloud provider to ever access user data or search queries.

CipherLocker shows that highly-scalable, fast *ranked* search on encrypted data is possible without the deployment of expensive and often insecure server-side search-on-encrypted-data cryptography which would require 3-5 orders of magnitude more resources and cannot scale to even thousands of users, or the simplest sharing scenarios without breaking security.

CipherLocker is the result of several years of work and it cannot be exhaustively detailed and analyzed in this space. This is the first of a series of papers discussing CipherLocker design, implementation and security properties. The main goal here is to briefly overview and introduce key design decisions and behaviors.

1 INTRODUCTION

Recent events [2, 7–13, 17] make it abundantly clear that now more than ever we are in dire need for provably-secure technologies that can enforce and protect personal rights and privacy.

CipherLocker is a new cross-platform (Linux, Mac, Windows), drag-and-drop encrypted file sharing platform designed and built from the ground up under a trust-nothing paradigm. As shown in

Figure 1: CipherLocker modules. All critical functionality is implemented client-side, including indexing (IDX) and searching (SRCH). **Client data is encrypted before being sent to the server. Keys are never shared with the server.** The server stores encrypted data (DataStore) and metadata to help clients synchronize and share (MDStore). The client transfer module (TM) interacts with the server messaging service (MS) for secure data sharing. The server access control module (ACM) validates clients and API requests.

Figure 1, all data is encrypted at origin (e.g., client laptop). Cloud-hosted data and API servers are not provided any encryption keys.

Unlike other efforts, e.g., [1, 3–6], CipherLocker does not allow any circumstances under which the server can get access to a user's keys, including in mobile settings (no "off-loading of decryption" to the cloud) or through a web interface (no decryption key release to any web server). Further, no weak cryptography is allowed. The server cannot force clients into compromised cipher settings. A simple, easy-to-use yet powerful, transparently integrated PKI-based trust layer enables users to build communities of trust; the design specifically defeats impersonation attacks by state-level adversaries which plague other existing efforts.

Full drag-and-drop OS integration makes things simple and easy to use. Sharing is achieved by a simple right-click on a folder, and adding a trusted share participant or group.

Encrypted Index Distribution. Client-Side Search. Finally, CipherLocker features fast ranked search on the encrypted data, 3-5 orders of magnitude faster than existing search-on-encrypted-data

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

CCS '18, October 15–19, 2018, Toronto, ON, Canada

© 2018 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-5693-0/18/10.

<https://doi.org/10.1145/3243734.3278500>

mechanisms. Multi-keyword ranked search can be executed in milliseconds and is based on a simple yet powerful client-side search and on-demand encrypted index distribution mechanism.

This is possible because fundamentally, for an overwhelming majority of data types, complex ranked indexing metadata can be represented and stored efficiently in 0.17-0.8% of the indexed data set. The current index-to-data ratio across all CipherLocker users is 0.0071. Further, paradoxically, index metadata growth slows down with larger data sets. This is due to data duplication and the increasingly media-intensive profiles of large data sets. As a result, for example, even an entry-level smartphone client with 8-32GB of local storage can comfortably index tens of TB of files.

Overall, CipherLocker guarantees that even if state-level adversaries fully compromise the storage and API servers no data leaks or user impersonation can occur.

Leaks. At most, the data server can observe: the structure of the sharing network, the file types (directory vs. file) and file hierarchy (e.g., directory *D* contains 5 files, and a subdirectory), block compression ratios (compression can be disabled by the user), the index-to-data ratios and data sync patterns (but not searches).

To avoid sync pattern leaks, deploying strong ORAM [15, 16] protocols is a theoretical solution but degrades performance beyond usability for most applications. Further, the increase in server load (especially random access, TCP connections, and the impossibility of deploying a scale-out object store as the data backend) results in an up to 6-7x higher cost per unit of data/time.

Open Source. We are working out the final licensing details and believe full openness and access to source code is essential for good security. CipherLocker has grown out of our internal need to collaborate and share under trust-nothing conditions. We have been using CipherLocker in-house for over 3 years now and are looking forward to inviting the greater community to participate in its development, design and use.

2 OVERVIEW

Figure 2: CipherLocker architecture. Horizontal bars = main layers, vertical bars = main modules (see Figure 1).

Shares and Permissions. Users can share some or all of their files with other parties and replicate/sync all or some of their files (including files in shares they are part of) across multiple devices.

The unit of sharing in CipherLocker is a folder. Once shared it is called a *share*. Shares have associated permissions including read, write, and re-share permissions.

Root Share. The CipherLocker file hierarchy starts in a special folder called the “root share”. CipherLocker does not access any files or folders residing on the local client outside of the root share.

The root share contains all the private user files and anything placed therein is synchronized into the cloud (and to all the other active clients of this user). Further, the root share folder will also contain the other shares this user is participating in.

Users can mark folders as “shared”; when doing so, a user becomes the first share participant, or *share owner*. Share owners can add other participants (users or groups) and grant them permissions.

Groups. Users can create *groups* which consist of a *group owner*, and a list of members. Recursive membership (groups can be part of other groups) is also possible (this functionality is not exposed in the client yet).

Trusted User List. Two users may securely share files only after exchanging and verifying public keys – at which point they each become part of the other’s *authenticated user list* (AUL). Key exchange is automatic. Key verification is also automatic – if both users are part of the same circle of trust, or semi-manual through a simple public key fingerprint verification otherwise.

System Modules. The client-side indexing (IDX) module (Figure 1) builds indices over all the files accessible to the client, and stores them in IndexDB. The Search (SRCH) module uses IndexDB to provide efficient search. The File System Watchdog (FSW) observes and timely reports file data and meta-data changes (within a client’s root share), e.g., file edit, deletion, move. The FSW en-queues detected changes for processing by workers part of the Transfer Module (TM). The Encryption and Integrity (EnI) module encrypts all the blocks (data and index) sent by the client to the server, then decrypts and verifies the integrity of data received from the server.

The client-side MetaData Repository (MDRepo) contains information about accessible shares and groups that this user is a member of. For each such share, MDRepo contains the list of all the users and groups, as well as their associated share permissions. For each group, MDRepo contains the list of all its members, and all the shares the group is part of and under what permissions. For each share, MDRepo contains the list of files, along with their current synchronization state on the local client. Finally, MDRepo also contains the list of users and associated public keys that are considered authenticated (the AUL).

The client-side Inbound Event Handler and the server-side Messaging Service (MS) form the notification system. They enable clients to download updates to accessible data, made from different clients installed on other machines. The APIServer is the central provider-side module, that interfaces with clients in order to store, synchronize and share, user data and indices.

Layers. CipherLocker is logically structured as a set of inter-connected logical layers spanned by a number of system modules (Figure 2).

	CipherLocker	Existing Systems				
Search In Encrypted Files Without Downloading	●	✗	✗	✗	✗	✗
Cloud Absolutely Never Sees Your Data	●	✗	✗	⊙	⊙	⊙
Cloud Never Sees Encryption Keys	●	✗	✗	⊙	⊙	⊙
Secure Public Key Exchange	●	✗	✗	✗	✗	✗
Share-Wide Integrity Protection	●	✗	✗	✗	✗	✗
Sharing Secure Against Cloud Insiders	●	✗	✗	✗	✗	✗
Grouping/Teaming Secure Against Cloud Insiders	●	✗	✗	✗	✗	✗
Source-Code Available for Evaluation	●	✗	✗	✗	✗	✗
Cryptographic Protocols Publicly Documented	●	✗	✗	⊙	⊙	⊙
Ranked Search	●	●	●	✗	✗	✗
Semantic Security	●	✗	✗	⊙	⊙	⊙
Zero-Leak	●	✗	✗	⊙	⊙	⊙
Redundancy Across Multiple Clouds	●	✗	✗	✗	✗	⊙

Figure 3: Succinct comparison with existing systems.

The *installation and trust model layer* handles trustworthy client installation, initialization and update issues. The *user trust layer* enables users to build communities of trust. The *event based messaging layer* provides the inter-client and client-to-server communications mechanisms. It defines and distributes system events with integrity assurances.

The *data layer* efficiently organizes the data outsourced by users, the *indexing layer* builds and maintains client-side indices over accessible data, and the *search layer* is responsible for performing user issued multi-keyword ranked searches over that data. The *file system layer* detects local data changes and updates the server. The *secure sharing layer* keeps user data synchronized across multiple sharing users and the devices they use.

3 CONCLUSIONS

We introduced CipherLocker, a practical, easy-to-use, client-side encrypted file sharing system with integrated ranked search. All user data and metadata is strongly encrypted before leaving the client, and the service provider is never allowed access to encryption keys, user data or search queries. Users can securely store, share, sync, and search the data. CipherLocker shows that highly-scalable, fast *ranked* search on encrypted data is possible without the deployment of expensive and often insecure server-side search-on-encrypted-data cryptography.

4 ACKNOWLEDGMENTS

We would like to thank the National Science Foundation who funded part of this work under TTP award 1562376.

REFERENCES

- [1] Bitcasa. <https://www.bitcasa.com>.
- [2] National Security Letters Statistics. <https://epic.org/privacy/nsll/>.
- [3] pcloud. <https://www.pcloud.com/>.
- [4] Spideroak. <https://spideroak.com>.
- [5] Teamdrive. <https://www.teamdrive.com>.
- [6] Tresorit. <https://tresorit.com>.
- [7] Yahoo data breaches. https://en.wikipedia.org/wiki/Yahoo_data_breaches.
- [8] CHECK POINT RESEARCH TEAM. More Than 1 Million Google Accounts Breached by Gooligan. <http://blog.checkpoint.com/2016/11/30/1-million-google-accounts-breached-gooligan/>.
- [9] MUNCASTER, P. Google Hit by Insider Data Breach. <https://www.infosecurity-magazine.com/news/google-hit-by-insider-data-breach/>.
- [10] SALCEDO, H. Security Breaches Show Box Not Secure for Small Businesses. <https://psg.hitachi-solutions.com/credeon/blog/security-breaches-show-box-not-secure-for-small-business>, February 2015.
- [11] SPRING, T. Box.com Plugs Account Data Leakage Flaw. <https://threatpost.com/box-com-plugs-account-data-leakage-flaw/122810/>, January 2017.
- [12] TECHNOLOGY NEWS. Facebook admits year-long data breach exposed 6 million users. <http://www.reuters.com/article/net-us-facebook-security-idUSBRE95K18Y20130621>, June 2013.
- [13] THE GUARDIAN. Dropbox hack leads to leaking of 68m user passwords on the internet. <https://www.theguardian.com/technology/2016/aug/31/dropbox-hack-passwords-68m-data-breach>, August 2016.
- [14] THE GUARDIAN. 'Extreme surveillance' becomes UK law with barely a whimper. <https://www.theguardian.com/world/2016/nov/19/extreme-surveillance-becomes-uk-law-with-barely-a-whimper>, November 2016.
- [15] WILLIAMS, P., SION, R., AND CARBUNAR, B. Building castles out of mud: Practical access pattern privacy and correctness on untrusted storage. In *Proceedings of the 15th ACM Conference on Computer and Communications Security* (New York, NY, USA, 2008), CCS '08, ACM, pp. 139–148.
- [16] WILLIAMS, P., SION, R., AND TOMESCU, A. Privatefs: A parallel oblivious file system. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security* (New York, NY, USA, 2012), CCS '12, ACM, pp. 977–988.
- [17] WIRED. Facebook and Netflix reset passwords after data breaches. <http://www.wired.co.uk/article/facebook-netflix-password-reset>.